Rebound Distinguishers: Results on the Full Whirlpool Compression Function

Mario Lamberger[1], Florian Mendel[1], Christian Rechberger[1], Vincent Rijmen[1,2,3], and Martin Schläffer[1]

[1] Institute for Applied Information Processing and Communications Graz University of Technology, Inffeldgasse 16a, A–8010 Graz, Austria.
[2] Department of Electrical Engineering ESAT/COSIC, Katholieke Universiteit Leuven. Kasteelpark Arenberg 10, B–3001 Heverlee, Belgium.
[3] Interdisciplinary Institute for BroadBand Technology (IBBT), Belgium.
mario.lamberger@iaik.tugraz.at

Abstract. Whirlpool is a hash function based on a block cipher that can be seen as a scaled up variant of the AES. The main difference is the (compared to AES) extremely conservative key schedule. In this work, we present a distinguishing attack on the full compression function of Whirlpool. We obtain this result by improving the rebound attack on reduced Whirlpool with two new techniques. First, the inbound phase of the rebound attack is extended by up to two rounds using the available degrees of freedom of the key schedule. This results in a near-collision attack on 9.5 rounds of the compression function of Whirlpool with a complexity of 2^{176} and negligible memory requirements. Second, we show how to turn this near-collision attack into a distinguishing attack for the full 10 round compression function of Whirlpool. This is the first result on the full Whirlpool compression function.

Keywords: hash functions, cryptanalysis, near-collision, distinguisher

1 Introduction

In the last few years the cryptanalysis of hash functions has become an important topic within the cryptographic community. Especially the collision attacks on the MD4 family of hash functions (MD4, MD5, SHA-1) have weakened the security assumptions of these commonly used hash functions [6, 7, 17, 24–26]. Still, most of the existing cryptanalytic work has been published for this particular family of hash functions. Therefore, the analysis of alternative hash functions is of great interest. In this article, we will present a security analysis of the Whirlpool hash function with respect to collision resistance.

Whirlpool is the only hash function standardized by ISO/IEC 10118-3:2004 (since 2000) that does not follow the MD4 design strategy. Furthermore, it has been evaluated and approved by NESSIE [20]. Whirlpool is commonly considered to be a conservative block-cipher based design with an extremely conservative key schedule and follows the wide-trail design strategy [4, 5]. Since its proposal in 2000, only a few results have been published.

Table 1. Summary of results for Whirlpool. Complexities are given in compression function evaluations, a memory unit refers to a state (512 bits). The complexities in brackets refer to modified attacks using a precomputed table taking 2^{128} time/memory to set up.

target	rounds	complexity runtime/memory	type	source
block cipher W	6	$2^{120}/2^{120}$	distinguisher	Knudsen [11]
hash function	4.5	$2^{120}/2^{7}$	collision	
hash function	6.5	$2^{128}/2^{7}$	near-collision	Mendel *et al.*
compression function	5.5	$2^{120}/2^{7}$	collision	FSE 2009 [16]
compression function	7.5	$2^{128}/2^{7}$	near-collision	
hash function	5.5	$2^{120+s}/2^{64-s}$	collision	Appendix A
hash function	7.5	$2^{128+s}/2^{64-s}$	near-collision	Appendix A
compression function	7.5	$2^{184}/2^{8}$ $(2^{120}/2^{128})$	collision	Sect. 4
compression function	9.5	$2^{176}/2^{8}$ $(2^{112}/2^{128})$	near-collision	Sect. 4
compression function	10	$2^{188}/2^{8}$ $(2^{121}/2^{128})$	distinguisher	Sect. 5

Related Work. At FSE 2009, Mendel *et al.* proposed a new technique for the analysis of hash functions: the *rebound attack* [16]. It can be applied to both block cipher based and permutation based constructions. The idea of the rebound attack is to divide an attack into two phases, an inbound and an outbound phase. In the inbound phase, degrees of freedom are used, such that in the outbound phase several rounds can be bypassed in both forward- and backwards direction. This led to successful attacks on round-reduced Whirlpool for up to 7.5 (out of 10) rounds. The results are summarized in Table 1.

For the block cipher W that is implicitly used in the Whirlpool compression function, Knudsen described an integral distinguisher for 6 out of 10 rounds [11]. Furthermore, it is assumed that this property may extend also to 7 rounds. Note that in [12] similar techniques were used to obtain known-key distinguishers for 7-rounds of the AES.

Our Contribution. The main contribution of this paper is a distinguishing attack on the full compression function of Whirlpool which is achieved by improving upon the work of Mendel *et al.* in [16] in several ways.

We start with a description of the hash function Whirlpool. Then, in Sect. 3, we give an overview of the rebound attack and show how it is applied to reduced versions of Whirlpool. In Sect. 4, we describe our improvement of the rebound attack on Whirlpool in detail. This technique enables us to add two rounds in the inbound phase of the attack and thus gives a collision and near-collision attack on the Whirlpool compression function reduced to 7.5 and 9.5 rounds, respectively. Based on this, we describe in Sect. 5 a new generic attack and show how to distinguish the full (all 10 rounds) compression function of Whirlpool from a random function by turning the near-collision attack for 9.5 rounds into a distinguishing attack for 10 rounds. To the best of our knowledge this is the

first result on the full Whirlpool compression function. Table 1 summarizes the previous results on Whirlpool as well as the contributions of this paper.

2 Description of Whirlpool

Whirlpool is a cryptographic hash function designed by Barreto and Rijmen in 2000 [1]. It is an iterative hash function based on the Merkle-Damgård design principle (*cf.* [18]). It processes 512-bit message blocks and produces a 512-bit hash value. If the message length is not a multiple of 512, an unambiguous padding method is applied. For the description of the padding method we refer to [1]. Let $M = M_1 \| M_2 \| \cdots \| M_t$ be a t-block message (after padding). The hash value $h = H(M)$ is computed as follows:

$$H_0 = IV \tag{1}$$

$$H_j = W(H_{j-1}, M_j) \oplus H_{j-1} \oplus M_j \quad \text{for } 0 < j \leq t \tag{2}$$

$$h = H_t \tag{3}$$

where IV is a predefined initial value and W is a 512 bit block cipher used in the Miyaguchi-Preneel mode [18]. The block cipher W used by Whirlpool is very similar to the Advanced Encryption Standard (AES) [19].

The state update transformation and the key schedule update an 8×8 state S and K of 64 bytes in 10 rounds. In one round, the state is updated by the round transformation r_i as follows:

$$r_i \equiv \mathsf{AK} \circ \mathsf{MR} \circ \mathsf{SC} \circ \mathsf{SB}.$$

The round transformations are briefly described here:

- the non-linear layer SubBytes (SB) applies an S-Box to each byte of the state independently.
- the cyclical permutation ShiftColumns (SC) rotates the bytes of column j downwards by j positions.
- the linear diffusion layer MixRows (MR) is a right-multiplication by the 8×8 circulant MDS matrix $cir(1, 1, 4, 1, 8, 5, 2, 9)$.
- the key addition `AddRoundKey` (AK) adds the round key K_i to the 8×8 state, and `AddConstant` (AC) adds the round constant C_i to the 8×8 state of the key schedule.

After the last round of the state update transformation, the initial value or previous chaining value H_{j-1}, the message block M_j, and the output value of the last round are combined (xored), resulting in the output of one iteration. A detailed description of the hash function is given in [1].

We denote the resulting state of round transformation r_i by S_i and the intermediate states after SubBytes by S_i^{SB}, after ShiftColumns by S_i^{SC} and after MixRows by S_i^{MR}. The initial state prior to the first round is denoted by $S_0 = M_j \oplus K_0$. The same notation is used for the key schedule with round keys K_i with $K_0 = H_{j-1}$.

3 The Rebound Attack

The rebound attack is a new tool for the cryptanalysis of hash functions and was published by Mendel *et al.* in [16]. It is a differential attack. The main idea is to use the available degrees of freedom in a collision attack to efficiently fulfill the low probability parts in the middle of a differential trail. The rebound attack consists of an inbound phase with a meet-in-the-middle part in order to exploit the available degrees of freedom, and a subsequent probabilistic outbound phase. AES based hash functions are a natural target for this attack, since their construction principle allows a simple application of the idea.

3.1 Basic Attack Strategy

In the rebound attack, the compression function, internal block cipher or permutation of a hash function is split into three sub-parts. Let W be a block cipher, then $W = W_{fw} \circ W_{in} \circ W_{bw}$.

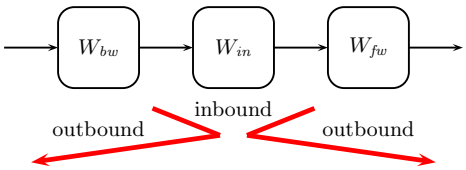

Fig. 1. A schematic view of the rebound attack. The attack consists of an inbound and two outbound phases.

The rebound attack can be described by two phases (see Fig. 1):

- **Inbound phase**: Is a meet-in-the-middle phase in W_{in}, which is aided by the degrees of freedom that are available to a hash function cryptanalyst. This very efficient combination of meet-in-the-middle techniques with the exploitation of available degrees of freedom is called the **match-in-the-middle approach**.
- **Outbound phase**: In the second phase, the matches of the inbound phase are computed in both forward- and backward direction through W_{fw} and W_{bw} to obtain desired collisions or near-collisions. If the differential trail through W_{fw} and W_{bw} has a low probability, one has to repeat the inbound phase to obtain more starting points for the outbound phase.

3.2 Preliminaries for the Rebound Attack on Whirlpool

In the following, we want to briefly summarize some well known facts that will be frequently used in the subsequent sections.

– *Truncated differentials*: Knudsen [10] proposed truncated differentials as a tool in block cipher cryptanalysis. In a *standard* differential attack (*cf.* [2]), the full difference between two inputs/outputs is considered whereas in the case of truncated differentials, the differences is only partially determined, *i.e.* for every byte, we only check if there is a difference or not. A byte having a non-zero difference is called *active*.
– *Difference Propagation in* MixRows: Since the MixRows operation is a linear transformation, standard differences propagate through MixRows in a deterministic way whereas truncated differences behave in a probabilistic way. The MDS property of the MixRows transformation ensures that the sum of the number of active input and output bytes is at least 9 (*cf.* [1]). In general, the probability of any $x \to y$ transition with $1 \leq x, y \leq 8$ satisfying $x+y \geq 9$ is approximately $2^{(y-8)\cdot 8}$. For a detailed description of the propagation of truncated differences in MixRows we refer to [16], see also [21].
– *Differential Properties of* SubBytes: Let $a, b \in \{0,1\}^8$. For the Whirlpool S-box, we are interested in the number of solutions to the equation

$$S(x) \oplus S(x \oplus a) = b. \tag{4}$$

Exhaustively counting over all 2^{16} differentials shows that the number of solutions to (4) can only be $0, 2, 4, 6, 8$ and 256, which occur with frequency $39655, 20018, 5043, 740, 79$ and 1, respectively. The task to return all solutions x to (4) for a given differential (a, b) is best solved by setting up a precomputed table of size 256×256 which stores the solutions (if there are any) for each (a, b).
However, it is easy to see that for any permutation S (to be more precise, for any injective map) the expected number of solutions to (4) is always 1. We get that $2^{-16} \sum_a \sum_b \#\{x \mid S(x \oplus a) \oplus S(x) = b\} = 2^{-16} \sum_a 2^8 = 1$, because for a fixed a, every solution x belongs to a unique b. Since the inputs to all the S-boxes are independent, the same reasoning is valid for the full SubBytes transformation.

3.3 Application to Round-Reduced Whirlpool

In this section, we will briefly describe the application of the rebound attack to the hash function Whirlpool. A detailed description of the attack is given in [16]. For a good understanding of our results, it is recommended to study these previous results on Whirlpool very carefully.

The rebound attack on Whirlpool is a differential attack which uses a differential trail with the minimum number of active S-boxes according to the wide trail design strategy. The core of the rebound attack on Whirlpool is a 4 round differential trail, where the fully active state is placed in the middle:

$$1 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 8 \xrightarrow{r_4} 1$$

In the rebound attack, one first splits the block cipher W into three subciphers $W = W_{fw} \circ W_{in} \circ W_{bw}$, such that the most expensive part of the differential trail is covered by the inbound phase W_{in}. In the inbound phase, the

available degrees of freedom (in terms of actual values of the state) are used to guarantee that the differential trail in W_{in} holds. The differential trail in the outbound phase (W_{fw}, W_{bw}) is supposed to have a relatively high probability. While standard XOR differences are used in the inbound phase, truncated differentials are used in the outbound phase of the attack.

In the following, we briefly describe the inbound and outbound phase of the rebound attack on 4 rounds of Whirlpool. For a more detailed description, we refer to the original paper [16].

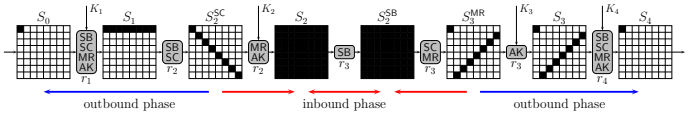

Fig. 2. A schematic view of the rebound attack on 4 rounds of Whirlpool with round key inputs. Black state bytes are active.

Inbound Phase. In the first step of the inbound phase, we choose a random difference with 8 active bytes at the input of MixRows of round r_2 (S_2^{SC}). Note that we need an active byte in each row of the state (see Fig. 2) to get a fully active state after the MixRows transformation. Since AddRoundKey does not change the difference, we get a fully active state at the input of SubBytes of round r_3 (S_2). Then, we start with another difference in 8 active bytes at the output of MixRows of round r_3 (S_3^{MR}) and propagate backwards. Again, since we have an active byte in each row, we get a fully active state at the output of SubBytes of round r_3.

In the second step of the inbound phase, the match-in-the-middle step, we look for a matching input/output difference of the SubBytes layer of round r_3. This is done as described in Sect. 3.2 with a precomputed 256×256 lookup table. Note that we can repeat the inbound phase at most about 2^{128} times. As indicated in Sect. 3.2, we expect one solution per trial, that is, we can produce at most 2^{128} actual values that follow the differential trail in the inbound phase.

Outbound Phase. In contrast to the inbound phase, we use truncated differentials in the outbound phase of the attack. By propagating the matching differences and state values through the next SubBytes layer outwards, we get a truncated differential in 8 active bytes in both backward and forward direction. These truncated differentials need to propagate from 8 to 1 active byte through the MixRows transformation, both in the backward and forward direction (see Fig. 2). The propagation of truncated differentials through the MixRows transformation can be modelled in a probabilistic way, see Sect. 3.2. Since we need to fulfill one $8 \rightarrow 1$ transitions in the backward and forward direction, the probability of the outbound phase is $2^{-2 \cdot 56} = 2^{-112}$. In other words, we have to

repeat the inbound phase about 2^{112} times to generate 2^{112} starting points for the outbound phase of the attack.

3.4 Previous Results on Round-Reduced Whirlpool

Extending the 4 round trail in both, the inbound and outbound phase, leads to attacks on round reduced Whirlpool for up to 7.5 (out of 10) rounds (where 0.5 rounds consist only of SubBytes and ShiftColumns). To be more precise, by extending the outbound phase of the attack by 0.5 and 2.5 rounds, one can construct a collision and near-collision for the Whirlpool hash function reduced to 4.5 and 6.5 rounds, respectively. The collision attack has a complexity of about 2^{120} and the near-collision attack has a complexity of about 2^{128}. Furthermore, by additionally extending the inbound phase of the attack by 1 round, one can find a collision and a near-collision for the compression function of Whirlpool reduced to 5.5 and 7.5 rounds with a complexity of 2^{120} and 2^{128}, respectively. Note that adding this round in the inbound phase is possible, since in a compression function attack, one can use the degrees of freedom of the key schedule (chaining value) to guarantee that the trail in the inbound phase holds. All results are summarized in Table 1 and for more details on these results we refer to [16].

4 Improved Rebound Attack on the Whirlpool Compression Function

In this section, we improve the inbound phase of the original rebound attack on Whirlpool. By using a new differential trail and extensively using the available degrees of freedom of the key schedule, we can add 2 additional rounds to the inbound phase of the attacks. The basic idea is to have two instead of one inbound phase (match-in-the-middle step) and connect them using the available degrees of freedom from the key schedule. The outbound phase of the attacks is identical as in the previous attacks on 5.5 and 7.5 rounds for the compression function of Whirlpool. As a result, we obtain a collision and a near-collision attack for the compression function of Whirlpool reduced 7.5 and 9.5 rounds, respectively.

4.1 Inbound Phase

In this section, we describe the improved inbound phase of the attack in detail. We use the following sequence of active bytes:

$$8 \xrightarrow{r_1} 64 \xrightarrow{r_2} 8 \xrightarrow{r_3} 8 \xrightarrow{r_4} 64 \xrightarrow{r_5} 8$$

In order to find inputs following the differential of the inbound phase, we split it into two parts. In the first part, we apply the match-in-the-middle step with active bytes $8 \rightarrow 64 \rightarrow 8$ twice in rounds 1-2 and 4-5. In the second part, we need to connect the resulting 8 active bytes and 64 (byte) values of the state between round 2 and 4 using the degrees of freedom we have in the choice of the round key values (see Fig. 3).

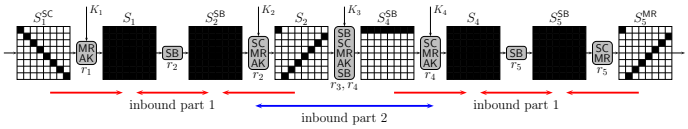

Fig. 3. The inbound phase of the attack.

Inbound Part 1. In this part of the inbound phase, we apply the match-in-the-middle step twice for rounds 1-2 and 4-5 (see Fig. 3), which can be summarized as follows:

1. Precomputation: For the S-box, compute a 256×256 lookup table as described in Sect. 3.2.
2. Match-in-the-middle (rounds 1-2):
   (a) Start with 8 active bytes at the output of AddRoundKey in round r_2 (S_2) and propagate backward to the output of SubBytes in round r_2 (S_2^{SB}).
   (b) Start with 8 active bytes at the input of MixRows in round r_1 (S_1^{SC}) and propagate forward to the input of SubBytes in round r_2 (S_1). Note that we can compute forward and solve the following step for each row independently.
   (c) Connect the input and output of the S-boxes of round r_2 by choosing the actual values of the state S_1, respectively S_2^{SB}, using the lookup table generated in the precomputation step. After repeating step (b) for each row about 2^8 times we expect to find a match for the 8 S-boxes and thus 2^8 actual values (see Sect. 3.2). Since we do this for all rows independently, we get about 2^{64} actual values for the full state S_1, respectively S_2^{SB}, such that the trail holds.
3. Match-in-the-middle (rounds 4-5): Do the same as in Step 2.

Hence, we get 2^{64} candidates for S_2^{SB} and 2^{64} candidates for S_4 after the first part of the inbound phase of the attack with a complexity of about 2^9 round transformations.

Inbound Part 2. In the second part of the inbound phase, we have to connect the 8 active bytes (64 (bit) conditions) as well as the actual values (512 (bit) conditions) of S_2^{SB} and S_4 by choosing the subkeys K_2, K_3 and K_4 accordingly. Therefore, we have to solve the following equation:

$$\mathsf{MR}(\mathsf{SC}(\mathsf{SB}(\mathsf{MR}(\mathsf{SC}(\mathsf{SB}(\mathsf{MR}(\mathsf{SC}(S_2^{\mathsf{SB}})) \oplus K_2))) \oplus K_3))) \oplus K_4 = S_4 \tag{5}$$

with

$$\begin{aligned} K_3 &= \mathsf{MR}(\mathsf{SC}(\mathsf{SB}(K_2))) \oplus C_3 \\ K_4 &= \mathsf{MR}(\mathsf{SC}(\mathsf{SB}(K_3))) \oplus C_4. \end{aligned} \tag{6}$$

Since we have 2^{64} candidates for S_2^{SB}, 2^{64} candidates for S_4 and 2^{512} candidates for the 3 subkeys K_2, K_3, K_4 (because of (6)), we expect to find 2^{64} solutions.

Since $S_2^{\mathsf{MR}} = \mathsf{MR}(\mathsf{SC}(S_2^{\mathsf{SB}}))$, we can rewrite the above equation as follows:

$$\mathsf{MR}(\mathsf{SC}(\mathsf{SB}(\mathsf{MR}(\mathsf{SC}(\mathsf{SB}(S_2^{\mathsf{MR}} \oplus K_2))) \oplus K_3))) \oplus K_4 = S_4 \tag{7}$$

Note that one can always change the order of SC and SB in the Whirlpool block cipher without affecting the output of one round. In order to make the subsequent description of the attack easier, we do this here and get the following equation.

$$\mathsf{MR}(\mathsf{SC}(\mathsf{SB}(\mathsf{MR}(\mathsf{SB}(\mathsf{SC}(S_2^{\mathsf{MR}} \oplus K_2))) \oplus K_3))) \oplus K_4 = S_4 \tag{8}$$

Furthermore, MR and SC are linear transformations and hence we can rewrite the above equation as follows:

$$\mathsf{SB}(\mathsf{MR}(\mathsf{SB}(S_2^* \oplus K_2^*)) \oplus K_3) \oplus K_4^{\mathsf{SB}} = X \tag{9}$$

with $S_2^* = \mathsf{SC}(S_2^{\mathsf{MR}})$, $K_2^* = \mathsf{SC}(K_2)$, $K_4^{\mathsf{SB}} = \mathsf{SB}(K_3)$, $X = \mathsf{SC}^{-1}(\mathsf{MR}^{-1}(S_4 \oplus C_4))$.

In the following, this equivalent description is used to connect the values and differences of the two states S_2^{MR} and S_4.

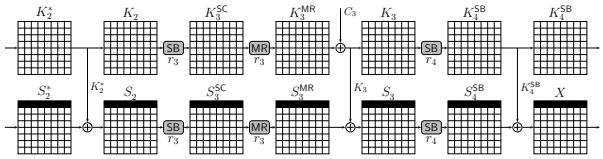

Fig. 4. The second part of the inbound phase. Black state bytes are active.

Remember that the two 8-byte differences of S_2^* and X have already been fixed due to the previous steps. Furthermore, we can choose from 2^{64} values for each of the states S_2^* and X. Now, we use equation (9) to determine the subkey K_2^* such that we get a solution for the inbound phase of the attack. Note that we can solve (9) for each row of the equation independently (see Fig. 4). It can be summarized as follows.

1. Compute the 8-byte difference and the 2^{64} values of the state S_2^* from S_2^{SB}, and compute the 8-byte difference and the 2^{64} values of the state X from S_4. Note that we can compute and store the values of S_2^* and X row-by-row and independently. Hence, both the complexity and memory requirements for this step are 2^8 instead of 2^{64}.
2. Repeat the following steps for all 2^{64} values of the first row of S_2 to get 2^{64} matches for S_2^* to S_4:
   (a) For the chosen value of the first row of S_2, forward compute the differences and values to the first row of S_3.
   (b) Choose the first row of the key K_3 such that the differential of the S-box between S_3 and S_4^{SB} holds.

(c) Compute the first row of K_2^*, S_2^*, K_4^{SB} and X. Since we have 2^{64} values for the first row of S_2^* and 2^{64} values for the first row of X, we expect to find a match on both sides. In other words, we have now connected the values and differences of the first row.

(d) Next, we connect the values of rows 2-8 independently by a simple brute-force search over all 2^{64} corresponding key values of K_2^*. Since we have to connect 64 bit values and we test 2^{64} key values we expect to always find a solution.

In total, we get 2^{64} matches connecting state S_2^* to state X with a complexity of 2^{128} and memory requirements of 2^8. In other words, with the values of S_2^*, X and the corresponding key K_2^*, we get 2^{64} starting points for the outbound phase of the attack. Hence, the average complexity to find one starting point for the outbound phase is 2^{64}. It is important to note that one can construct a total of 2^{192} starting points in the inbound phase to be used in the outbound phase of the attack.

Note that step 2 (d) can be implemented using a precomputed lookup table of size 2^{128}. In this lookup table each row of the key K_2 (64 bits) is saved for the corresponding two rows of S_2^* and X (64 bits each). Using this lookup table, we can find one starting point for the outbound phase with an average complexity of 1. However, the complexity to generate this lookup table is 2^{128}.

4.2 Outbound Phase

In the outbound phase of the attack, we further extend the differential path backward and forward. By propagating the matching differences and state values through the next SubBytes layer, we get a truncated differential in 8 active bytes for each direction. These truncated differentials need to follow a specific active byte pattern to result in a collision on 7.5 rounds and a near-collision on 9.5 rounds, respectively. In the following, we will describe the outbound phase for the collision and near-collision attack in detail.

Collision for 7.5 Rounds. By adding 1 round in the beginning and 1.5 rounds at the end of the trail, we get a collision for 7.5 rounds for the compression function of Whirlpool. In the attack, we use the following sequence of active bytes:

$$1 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 8 \xrightarrow{r_4} 8 \xrightarrow{r_5} 64 \xrightarrow{r_6} 8 \xrightarrow{r_7} 1 \xrightarrow{r_{7.5}} 1$$

As described in Sect. 3.2, the propagation of truncated differentials through the MixRows transformation is modelled in a probabilistic way. For the differential trail to hold, we need that the truncated differentials in the outbound phase propagate from 8 to 1 active byte through the MixRows transformation, both in the backward and forward direction (see Fig. 5). Since the transition from 8 active bytes to 1 active byte through the MixRows transformation has a probability of about 2^{-56}, the probability of this part of the outbound phase is

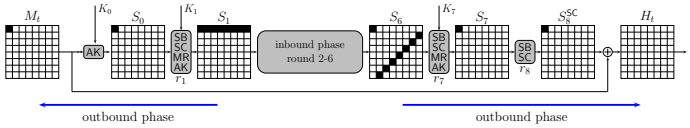

Fig. 5. Differential trail for collision attack on 7.5 rounds.

$2^{-2 \cdot 56} = 2^{-112}$. Furthermore, to construct a collision at the output (after the feed-forward), the exact value of the input and output difference has to match. Since only one byte is active (see Fig. 5), this can be fulfilled with a probability of 2^{-8}. Hence, the probability of the outbound phase is $2^{-112} \cdot 2^{-8} = 2^{-120}$. In other, words, we have to generate 2^{120} starting points (for the outbound phase) in the inbound phase of the attack to find a collision for the compression function of Whirlpool reduced to 7.5 rounds.

Since we can find one starting point with an average complexity of about 2^{64} and memory requirements of 2^8, we can find a collision with a complexity of about $2^{120+64} = 2^{184}$. The complexity of the attack can be further improved on the cost of higher memory requirements. By using a lookup table with 2^{128} entries (generated in a precomputation step), we can find one starting point for the inbound phase with an average complexity of 1. In other words, we can find a collision for the compression function reduced to 7.5 rounds with a complexity of about 2^{120}. However, the precomputation step (constructing the lookup table) has a complexity of about 2^{128}.

Near-Collision for 9.5 Rounds. The collision attack on 7.5 rounds for the compression function can be further extended by adding one round at the beginning and one round at the end of the trail in the outbound phase. The result is a near-collision attack on 9.5 rounds for the compression function of Whirlpool with the following sequence of active bytes:

$$8 \xrightarrow{r_1} 1 \xrightarrow{r_2} 8 \xrightarrow{r_3} 64 \xrightarrow{r_4} 8 \xrightarrow{r_5} 8 \xrightarrow{r_6} 64 \xrightarrow{r_7} 8 \xrightarrow{r_8} 1 \xrightarrow{r_9} 8 \xrightarrow{r_{9.5}} 8$$

Since the 1-byte difference at the beginning and end of the 7.5 round trail will always result in 8 active bytes after one MixRows transformation (see Sect. 3.2), we can go backward 1 round and forward 1 round with no additional cost. Using the feed-forward, the position of two active S-boxes match and cancel each other with a probability of 2^{-16}. Hence, we get a collision in 50 and 52 bytes for the compression function of Whirlpool with a complexity of about 2^{176} and $2^{176+16} = 2^{192}$, respectively. With a precomputation step with complexity of 2^{128} and similar memory requirement, one can find a near-collision for the compression function of Whirlpool with a complexity of about 2^{112} (collision in 50 bytes) and 2^{128} (collision in 52 bytes), respectively.

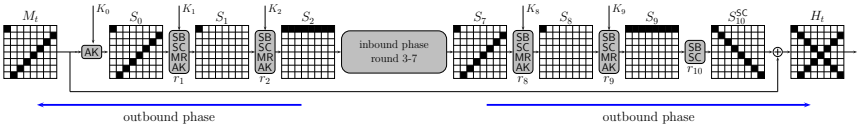

Fig. 6. In the attack on 9.5 rounds we extend the trail one more round at the beginning and at the end of the outbound phase to get a near-collision of Whirlpool.

5 A Subspace Distinguisher for 10 Rounds

In this section, we present the first cryptanalytic result on the full Whirlpool compression function. The method for extending the previous result on 9.5 rounds is extended to full 10 rounds of the compression function by defining a different attack scenario. Instead of aiming for a near-collision, we are interested in distinguishing the Whirlpool compression function from a random function. For this, we will introduce a new kind of distinguishing attack, a so called *subspace distinguisher*. In the following, $\mathbb{F}_2 = GF(2)$ always denotes the finite field of order 2.

For the subspace distinguishing attack, we consider the following problem:

Problem 1 *Given a function f mapping to $\mathbb{F}_2^N$, try to find t input pairs such that the corresponding output differences belong to a vector space of dimension at most n for some $n \leq N$.*

Remark. We define Problem 1 in this generic way in order to make it more generally applicable. This will be shown in the extended version of this paper.

5.1 Solving Problem 1 for the Whirlpool Compression Function

In this section, we show how the compression function attack described in Sect. 4 can be used to distinguish the full Whirlpool compression function from a random function.

Obviously, the difference between two Whirlpool states can be seen as a vector in the vector space of dimension $N = 512$ over $\mathbb{F}_2$. The crucial observation is that the attack of Sect. 4 can be interpreted as an algorithm that can find t difference vectors in $\mathbb{F}_2^{512}$ (output differences of the compression function) that form a vector space of dimension $n \leq 128$.

To see this, observe that by extending the differential trail from 9.5 to 10 rounds, the 8 active bytes in S_{10}^{SC} will always result in a fully active state S_{10} due to the properties of the MixRows transformation. Thus the near-collision is destroyed. However, if we look again at Fig. 6, the differences in M_t and the differences in S_{10}^{SC} can be seen as (difference) vectors belonging to subspaces of $\mathbb{F}_2^{512}$ of dimension at most 64.

Even though after the application of MixRows and AddRoundKey the state S_{10} is fully active in terms of truncated differentials, the differences in S_{10} still belong to a subspace of $\mathbb{F}_2^{512}$ of dimension at most 64 due to the properties of

MixRows. Therefore, after the feed-forward, we can conclude that the differences at the output of the compression function form a subspace of $\mathbb{F}_2^{512}$ of dimension $n \leq 128$.

Hence, we can use the attack of Sect. 4 to find t difference vectors forming a vector space of dimension $n \leq 128$ with a complexity of $t \cdot 2^{176}$ or $t \cdot 2^{112}$ using a precomputation step with complexity 2^{128}. Note that $t \leq 2^{192-112} = 2^{80}$ due to the remaining degrees of freedom in the inbound phase of the attack.

Now the main question is for which values of t our attack is more efficient than a generic attack. In other words, how do we have to choose t such that we can distinguish the compression function of Whirlpool from a random function. Therefore, we first have to bound the complexity of the generic attack. This is described in the next section.

5.2 Solving Problem 1 for a Random Function

Remarks on the Security Model. In order to discuss generic attack scenarios, we will have to choose a security model. We will adopt the black box model introduced by Shannon [23]. In this model, a block cipher can be seen as a family of functions parameterized by the secret key $k \in \mathcal{K}$, that is, $E : \{0,1\}^{|k|} \times \{0,1\}^N \mapsto \{0,1\}^N$, where for each $k \in \mathcal{K}$, E_k is seen as a uniformly chosen random permutation on $\{0,1\}^N$.

In [3] it was shown, that an ideal block cipher based hash function in the Miyaguchi-Preneel mode is collision resistant and non-invertible. Based on this, we model our compression function f as black box oracle to which only forward queries are admissible. We also want to note that in all of the following, when we are talking about complexity, we are talking about *query complexity*. Note that the practical complexity is always greater or equal to the query complexity.

The Generic Approach. In this generic approach the only property used about f is the fact that the outputs of f are contained in the vector space $\mathbb{F}_2^N$.

Let us now assume that an adversary is making Q queries to the function f. Assuming that $Q \ll 2^{N/2}$, we thus get $K = \binom{Q}{2}$ differences ($\in \mathbb{F}_2^N$) coming from these Q queries. For given n and $t \gg n$, we now want to calculate the probability that among these K difference vectors, we have t vectors that span a space of dimension less or equal to n.

We will need the following fact about matrices over finite fields. Let $E(t, N, d)$ denote the number of $t \times N$ matrices over $\mathbb{F}_2$ that have rank equal to d. Then, it is well known (*cf.* [9] or [13]) that

$$E(t, N, d) = \prod_{i=0}^{d-1} (2^N - 2^i) \cdot \binom{t}{d}_2 = \prod_{i=0}^{d-1} \frac{(2^N - 2^i) \cdot (2^t - 2^i)}{2^d - 2^i}, \qquad (10)$$

where $\binom{t}{d}_2$ denotes the q-binomial coefficient with $q = 2$.

Proposition 1 *Let $n, t, N \in \mathbb{N}$ be given such that $t \gg N > n$. We assume a set of K vectors chosen uniformly at random from $\mathbb{F}_2^N$. Let $\Pr(K, t, N, n)$ denote the probability that t of these K vectors span a space of dimension not larger than n. Then, we have*

$$\Pr(K, t, N, n) \leq \binom{K}{t} 2^{-t \cdot N} \sum_{d=1}^{n} E(t, N, d) \tag{11}$$

$$\leq \frac{1}{\sqrt{2\pi t}} \left(\frac{Ke}{t}\right)^t 2^{-(N-n)(t-n)-(n-1)}. \tag{12}$$

Proof. Based on the definition of $E(t, N, d)$, it is easy to see that (11) is an upper bound for $\Pr(K, t, N, n)$.

Computing the second bound consists of two steps. Bounding the binomial coefficient and bounding the rest. We get

$$2^{-t \cdot N} \sum_{d=1}^{n} E(t, N, d) \leq 2^{-t \cdot N} \cdot 2 \cdot E(t, N, n) \tag{13}$$

$$\leq 2^{-t \cdot N + 1} \left(\frac{(2^t - 2^{n-1}) \cdot (2^N - 2^{n-1})}{2^n - 2^{n-1}}\right)^n \tag{14}$$

$$\leq 2^{-t \cdot N + 1} \left(2^{n-1} \cdot 2^{t-(n-1)} \cdot 2^{N-(n-1)}\right)^n \tag{15}$$

$$= 2^{-(t-n)(N-n)-(n-1)}. \tag{16}$$

These inequalities are based on two facts. First, it is easy to show that for $t \gg N > n$, we have $E(t, N, n) \leq \sum_{d=1}^{n} E(t, N, d) \leq 2 \cdot E(t, N, n)$. This can be proven by using induction over n and elementary properties of the q-binomial coefficient. Second, (14) follows from the fact that the function defined by $f(x) = (2^t - x)(2^N - x)/(2^n - x)$ is strictly increasing on the interval $x \in [0, 2^{n-1}]$.

For the binomial coefficient $\binom{K}{t}$ we combine the simple estimate $\binom{K}{t} \leq K^t/t!$ with the following inequality based on Stirling's formula [22]:

$$\sqrt{2\pi} t^{t+\frac{1}{2}} e^{-t+\frac{1}{12t+1}} < t! < \sqrt{2\pi} t^{t+\frac{1}{2}} e^{-t+\frac{1}{12t}} \tag{17}$$

From this we get $\binom{K}{t} \leq \frac{1}{\sqrt{2\pi t}} \left(\frac{K \cdot e}{t}\right)^t$ and with (16), this proves the proposition. ∎

As a corollary, we can give a lower bound for the number of random vectors needed to fulfill the conditions of the proposition with a certain probability.

Corollary 1 *For a given probability p and given N, n, t as in Proposition 1, the number K of random vectors needed to contain t vectors spanning a space of dimension not larger than n with a probability p is lower bounded by*

$$K \geq \frac{1}{e} \left(p\sqrt{2\pi t}\right)^{\frac{1}{t}} \cdot t \cdot 2^{\frac{(N-n)(t-n)+(n-1)}{t}}. \tag{18}$$

and the number of queries Q to f needed to produce t vectors spanning a space of dimension not larger than n with a probability p is lower bounded by

$$Q \geq \sqrt{\frac{2}{e}} \left(p\sqrt{2\pi t}\right)^{\frac{1}{2t}} \cdot \sqrt{t} \cdot 2^{\frac{(N-n)(t-n)+(n-1)}{2t}}. \tag{19}$$

Proof. Equation (18) follows immediately from (12) and (19) follows from setting $K = \binom{Q}{2} = Q(Q-1)/2$ in (18). ∎

5.3 Complexity of the Distinguishing Attack

Table 2 shows the complexities of the generic approach and our dedicated approach for several values of t. As can be seen in the table, one can distinguish the full Whirlpool compression function from random with a complexity of about 2^{188} with $t = 2^{12}$ (or 2^{121} with $t = 2^9$ using a precomputation table). In other words, when performing 2^{188} queries to a random function (19) shows that the probability for solving Problem 1 for $t = 2^{12}$ is $\ll 1$. To the best of our knowledge this is the first result on the full Whirlpool compression function.

Table 2. Values for t, Q (query complexity), C (complexity of our attack), and C_p (complexity of our attack with precomputation) for $p = 1, N = 512, n = 128$

$\log_2(t)$	$\log_2(Q)$	$\log_2(C)$	$\log_2(C_p)$	$\log_2(t)$	$\log_2(Q)$	$\log_2(C)$	$\log_2(C_p)$
9	148.41	185	121	13	195.29	189	125
10	172.84	186	122	14	197.28	190	126
11	185.31	187	123	15	198.53	191	127
12	191.80	188	124	16	199.40	192	128

6 Conclusion

In this paper, we have proposed a new kind of distinguishing attack for cryptanalysis of hash functions. We have successfully attacked the Whirlpool compression function. To the best of our knowledge this is the first attack on full Whirlpool.

We have obtained this result by improving the rebound attack on reduced Whirlpool. First, the inbound phase of the rebound attack was extended by up to two rounds using the available degrees of freedom from the key schedule. This resulted in a near-collision attack on 9.5 rounds of the compression function of Whirlpool. Second, we have shown how to turn this rebound near-collision attack into a distinguishing attack for the full 10 round compression function of Whirlpool.

The idea seems applicable to a wider range of hash function constructions. In particular, the attacks described in this paper can be applied to the hash

function Maelstrom [8] in a straight forward manner because of the similarity to Whirlpool (see also [16]). Several SHA-3 candidates are a natural target for this new kind of attack, see for instance [14, 15]. Furthermore, subspace distinguishers can be applied to block ciphers as well. This will be discussed in an extended version of this paper.

Acknowledgements

The authors wish to thank the anonymous referees for useful comments and discussions. The work in this paper has been supported in part by the European Commission under contract ICT-2007-216646 (ECRYPT II) and in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

References

1. Barreto, P.S.L.M., Rijmen, V.: The Whirlpool Hashing Function. Submitted to NESSIE, September 2000, revised May 2003. Available online at `http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html` (2008/12/11)
2. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO. LNCS, vol. 537, pp. 2–21. Springer (1990)
3. Black, J., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO. LNCS, vol. 2442, pp. 320–335. Springer (2002)
4. Daemen, J., Rijmen, V.: The Wide Trail Design Strategy. In: Honary, B. (ed.) IMA Int. Conf. LNCS, vol. 2260, pp. 222–238. Springer (2001)
5. Daemen, J., Rijmen, V.: The Design of Rijndael. Information Security and Cryptography, Springer (2002), iSBN 3-540-42580-2
6. De Cannière, C., Mendel, F., Rechberger, C.: Collisions for 70-Step SHA-1: On the Full Cost of Collision Search. In: Adams, C.M., Miri, A., Wiener, M.J. (eds.) Selected Areas in Cryptography. LNCS, vol. 4876, pp. 56–73. Springer (2007)
7. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT. LNCS, vol. 4284, pp. 1–20. Springer (2006)
8. Filho, D.G., Barreto, P.S., Rijmen, V.: The Maelstrom-0 hash function. In: SBSeg 2006 (2006)
9. Fisher, S.D.: Classroom Notes: Matrices over a Finite Field. Amer. Math. Monthly 73(6), 639–641 (1966)
10. Knudsen, L.R.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) FSE. LNCS, vol. 1008, pp. 196–211. Springer (1994)
11. Knudsen, L.R.: Non-random properties of reduced-round Whirlpool. NESSIE public report, NES/DOC/UIB/WP5/017/1 (2002)
12. Knudsen, L.R., Rijmen, V.: Known-key distinguishers for some block ciphers. In: Kurosawa, K. (ed.) ASIACRYPT. LNCS, vol. 4833, pp. 315–324. Springer (2007)
13. Lidl, R., Niederreiter, H.: Finite Fields, Encyclopedia of Mathematics and its Applications, vol. 20. Cambridge University Press, Cambridge, second edn. (1997), with a foreword by P. M. Cohn

14. Matusiewicz, K., Naya-Plasencia, M., Nikolić, I., Sasaki, Y., Schläffer, M.: Rebound Attack on the Full LANE Compression Function. In: Matsui, M. (ed.) ASIACRYPT. LNCS, Springer (2009), to appear
15. Mendel, F., Peyrin, T., Rechberger, C., Schläffer, M.: Improved Cryptanalysis of the Reduced Grøstl Compression Function, ECHO Permutation and AES Block Cipher. In: Jacobson, Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) Selected Areas in Cryptography. LNCS, Springer (2009), to appear
16. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) FSE. LNCS, vol. 5665, pp. 260–276. Springer (2009)
17. Mendel, F., Rijmen, V.: Cryptanalysis of the Tiger Hash Function. In: Kurosawa, K. (ed.) ASIACRYPT. LNCS, vol. 4833, pp. 536–550. Springer (2007)
18. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1997), available online at `http://www.cacr.math.uwaterloo.ca/hac/`
19. National Institute of Standards and Technology: FIPS PUB 197, Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, U.S. Department of Commerce (November 2001)
20. NESSIE: New European Schemes for Signatures, Integrity, and Encryption. IST-1999-12324, available online at `http://cryptonessie.org/`
21. Peyrin, T.: Cryptanalysis of Grindahl. In: Kurosawa, K. (ed.) ASIACRYPT. LNCS, vol. 4833, pp. 551–567. Springer (2007)
22. Robbins, H.: A remark on Stirling's formula. Amer. Math. Monthly 62, 26–29 (1955)
23. Shannon, C.E.: Communication Theory of Secrecy Systems. Bell Systems Technical Journal 28, 656–715 (1949)
24. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT. LNCS, vol. 3494, pp. 1–18. Springer (2005)
25. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO. LNCS, vol. 3621, pp. 17–36. Springer (2005)
26. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT. LNCS, vol. 3494, pp. 19–35. Springer (2005)

A Attacks on the Hash Function

In this section, we present a collision and near-collision for the Whirlpool hash function. The attacks are a straight forward extension of the collision and near-collision attack on 4.5 and 6.5 rounds of Whirlpool presented in [16]. By adding one round in the inbound phase we can find a collision and a near-collision for Whirlpool reduced to 5.5 and 7.5 rounds, respectively. The core of the attack is a 5 round differential trail, where two fully active states are placed in the middle:

$$1 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 64 \xrightarrow{r_4} 8 \xrightarrow{r_5} 1$$

Since the outbound phase of the attacks is identical to the previous attacks (see Sect. 4), we only discuss the inbound phase of the attack here (see Fig. 7).
It can be summarized as follows.

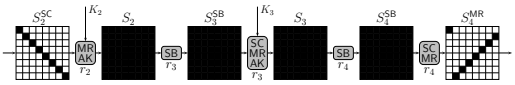

Fig. 7. The inbound phase of the collision attack and near-collision attack on the hash function.

1. Precomputation: For the S-box, compute a 256×256 lookup table as described in Sect. 3.2.
2. Start with 8 active bytes (differences) at the input of MixRows in round r_2 (S_2^{SC}) and propagate forward to the input of SubBytes in round r_3 (S_2).
3. Start with 8 active bytes at the output of MixRows in round r_4 (S_4^{MR}) and propagate backward to the output of SubBytes in round r_4 (S_4^{SB}).
4. Next we have to connect the states S_2 and S_4^{SB} such that the differential trail holds. In other words, we have to find the actual values for S_2 such that:

$$\mathsf{SB}(\mathsf{MR}(\mathsf{SC}(\mathsf{SB}(S_2))) \oplus K_3) \oplus \mathsf{SB}(\mathsf{MR}(\mathsf{SC}(\mathsf{SB}(S_2 \oplus \Delta_1))) \oplus K_3) = \Delta_2$$

   where Δ_1 denotes the active bytes (differences) in S_2 and Δ_2 denotes the active bytes (differences) in S_4^{SB}. In the following, we will show how this equation can be solved with a complexity of about 2^{64} by solving the equation for sets of 8 bytes independently. It can be summarized as follows.
   (a) For all 2^{64} values of $S_2[0,0], S_2[1,7], \ldots, S_2[7,1]$ compute the first row of S_4^{SB} and check if the above equation holds. Note that due to ShiftColumns, these bytes are shifted to the first row of S_3^{SC} and MixRows works on each row independently. In other words, we get 2^{64} candidates for each row of S_4^{SB}. Hence, after testing all 2^{64} candidates for the first row of S_4^{SB} we expect to find a match for the first row of Δ_2.
   (b) Do the same for the corresponding 8 bytes for row 2-8 of S_4^{SB}.
   After testing each set of 8 bytes independently, we will find a state S_2 such that the differential trail is connected. Finishing this step of the attack has a complexity of about $8 \cdot 2^{64}$ MixRows ($\approx 2^{64}$ round computations).

Hence, we can compute one starting point for the outbound phase with a complexity of about 2^{64}. Note that the complexity of the inbound phase can be significantly reduced at the cost of higher memory requirements. By saving 2^{64-s} candidates for S_4^{SB} in a list, we can do a standard time/memory tradeoff with a complexity of about 2^{120+s} and memory requirements of 2^{64-s}. By setting $s = 0$ we can find 2^{64} starting points with a complexity of 2^{64} and similar memory requirements of 2^{64}.

Hence, we can find a collision for Whirlpool reduced to 5.5 rounds with a complexity of about 2^{120} and a near-collision for 7.5 rounds in 50 (respectively 52) bytes with a complexity of about 2^{120} and 2^{112} (respectively 2^{128}). All attacks have memory requirements of 2^{64}.